Electronic Version
Stylesheet Version v1.1.1

Description

METHOD FOR SUCCESSIVE PLACEMENT BASED REFINEMENT OF A GENERALIZED COST FUNCTION

Background of Invention

[0001] The invention is generally related to the field of Electronic Design Automation as applied to the design of semiconductor chips, and more specifically, to a method for optimizing the placement of cells on the chip.

[0002] Placement has been an integral part of any VLSI chip design for many decades. Several approaches exist in the art for solving the placement problem. One of the most successful ones, known as partitioning, has been widely described in the technical literature, as for instance, by R.S. Tsay, E. S. Kuh and C.P. Hsu, "PROUD: a Sea-of-Gates Placement Algorithm", published in IEEE Design & Test of Computers, Vol. 5, Dec. 1988; by J. M. Kleinhans, G. Sigl and F. Johannes; "Gordian: A New Global Optimization / Rectangle Dissection Method for Cell Placement", pub-

lished in the Proceedings of the International Conference on Computer-Aided Design, pp. 506-509, 1988; and by G. Sigl, K. Doll and F. Johannes, "Analytical Placement: A Linear or Quadratic Objective Functions?", published in the Proceedings of the Design Automation Conference, pp. 427-432, 1991.

[0003] An input to the placement process consists of a VLSI chip physical footprint that includes positioning the I/O (input/output) ports that connect the chip to the external world. Additionally, a description of the chip area is selected in order to assign physical locations corresponding to the circuits forming the chip. A second input consists of a chip level netlist that describes all the circuits that are to be placed. The description of the circuit includes attributes such as size, pin location, power connection ports and their respective locations. The netlist file also includes a description of the cell connectivity. In most instances, the location attribute is empty, and it is for the placement process to assign the locations. In some instances, cells in the netlist are preplaced at fixed locations, in which case the placement tool is expected to honor the location attribute according to the input data. The placement process assigns locations to all the objects present in the in-

put netlist. The locations must be valid (i.e., legal) and non-overlapping, (i.e., the circuits must be placed in accordance to the description of the chip area -- one of the inputs). Many such placement algorithms meeting these criteria are already in existence.

[0004] An ideal placement process is one that optimizes the characteristics of the VLSI chip by generating a routable placement characterized by having a minimum total wire length, a minimum timing cost, and a minimum signal integrity cost, wherein the cost is defined as a function of the following parameters:

[0005] a) the total wire length consisting of the sum of the wire lengths over all the nets of the netlist. Individual net lengths are computed using Steiner and Net-Half Perimeter methods. The Steiner wire length of a net is computed by generating a Steiner routing pattern, i.e., one that connects all of the pins on a net using horizontal and vertical line segments. These line segments may intersect each other at points that do not correspond to net pins. Such points are known as Steiner points. The length is then computed after the Steiner pattern is generated by adding the sum of the lengths of each individual horizontal and vertical line segment. This value is then used to represent

the length of the net. The net-half-perimeter wire length refers to a method used for estimating the wire length. It involves creating a rectangle such that all the pins on the net remain within or at the boundary of the rectangle. Furthermore, the rectangle generated must be as small as possible and still satisfy all the aforementioned conditions. Finally, the perimeter of this rectangle is measured and halved, yielding a value known as the net-half-perimeter;

[0006] b) congestion, that defines the degree of routability of the design. Most commercial routers offer a global routing component providing this metric;

[0007] c) timing, that defines the intrinsic timing characteristics of the placement, computed by first repowering and buffering the design to be followed by a timing analysis; and

[0008] d) signal integrity, that qualifies the capacitive net coupling potential of the placement, and the probability of such coupling leading to functional and/or timing related problems.

[0009] Referring now to Figure 1 there is shown a representation of a conventional placement sequence.

[0010] Partitioning uses a "divide and conquer" approach to

achieve its objectives. Initially, and referring to Figure 1A, all the circuits are randomly placed on the chip regardless of the cost function optimization. In the case of a two-way partitioning (Figure 1B), the chip is subdivided in two regions. The process of subdivision will be referred hereinafter as a "cut". Starting from the rectangular outline of the entire chip, it is accomplished by drawing a vertical (or horizontal) line through the middle of the chip. The partitioning line together with the full chip outline defines the two regions. The next step consists in taking all the circuits falling within a window bounded by the two regions and assigns them to either region 1 or region 2 in a manner that minimizes the connectivity of the circuits in both regions. The remaining regions are then subdivided, as shown in Figures 1C-1E. The process proceeds recursively until each circuit is assigned a specific location in the chip. The process described thus far corresponds to the aforementioned two-way partitioning.

[0011] It is possible to perform two cuts at a time to achieve a four-way partitioning. The initial state of a four-way partitioning is based on circuits randomly placed on the chip that were previously shown in Figure 1A. The placement step that follows corresponds to Figure 1C, while the third

state corresponds to Figure 1E. This pattern continues until the end of the placement sequence is reached. The invention described herein applies to the general partitioning case that includes both, a two-way and a four-way partitioning.

[0012] In a more general way, a complete successive partitioning pass starts at cut 1 and terminates at cut N. Referring to Figure 1B, the placement shown therein results from cut number 1, while Figures 1C-1E depict the results from cuts number 2, 3, and 4 This pattern continues until cut number N is reached, corresponding to a fully placed netlist, wherein the number of partitions is sufficiently small to specify its detailed placement qualities. The value of N is computed by the following equation 1:

[0013] $N = \log_2 (\text{image\_area} / \text{smallest\_cell\_size})$ (1)

[0014] The image area variable in equation (1) is a measure of the amount of circuitry that is to be placed on the chip. The smallest_ cell_size variable is a measure of the area occupied by one occurrence of the cell. Dividing the image area by the smallest_ cell_size, the maximum number of cells that can possibly be placed on the chip is determined. By taking the $\log_2$ of this number, the number of two-way partitioning operations required to have all the

final partitions on the chip having only one cell is computed.

[0015] In previous generations of a VLSI design, placement was performed in a single stand-alone step, with a primary objective of generating a routable design (refer to the congestion cost metric described above) having a minimum total wire length. Since then, the technology trend has gravitated toward larger design sizes, concentrating rather on the chip performance determined by the interconnect delay, and which is considered be a stronger function of the placement solution. Thus, the importance of optimizing the placement beyond the total wire length and congestion has now acquired a new and higher level of importance. As a result, placement algorithms are now imbedded into timing closure systems (like PDS, Magma"s tools, Synopsys PD compiler, Cadence PKS, etc), where the placement itself is only one aspect of the optimization paradigm. These modern generation tools are designed to optimize the placement across many metrics, such as wire length, congestion, timing, and signal integrity.

[0016] Simultaneous optimization of the cost functions is difficult in view of certain cost functions competing against one another. By way of example, better timing may require

extra wire length, which is in direct conflict with the goal of minimizing the wire length. The same can be said with regard to congestion and wire length, i.e., reducing congestion by increasing total wire length. In addition to internal cost function conflicts there are also other important optimizations such as timing, power, and signal integrity that are not easily visible to a native placement algorithm. For these reasons, it becomes more difficult to achieve the optimal placement.

[0017] Accordingly, there is a need for a general framework to achieve an optimal placement that specifically addresses optimizations that include global qualities of the placement solution, defined as those related to the most significant organization of logic placed on the VLSI chip. This need is particularly important for VLSI chips that include a hierarchy of logic functions. For instance, individual circuits are normally handled at the leaf level of the hierarchy. Moving up the hierarchy, there is a coarsening of the structure that reflects larger and larger logic groupings. It is this relative placement of larger logic groupings that defines the global qualities of the placement. The relative ordering and specific placement of circuits within the "larger logic groupings" relates to the local quality of the

placement solution. It should be understood that the quality of the placement spans across a spectrum that includes global qualities on one end, and local qualities on the other. In the middle of the spectrum, there is a blend of global and local qualities.

[0018] Several approaches addressing congestion and timing optimization during placement are found in the literature. Techniques for optimization of congestion have been described, for instance, in:

[0019] U.S. Patent No. 6,068,662, Method and Apparatus for Congestion Removal, issued to Scepanovic, et al;

[0020] U.S. Patent No.6,075,933, Method and Apparatus for Continuous Column Density Optimization, issued to Pavisic, et al.;

[0021] U.S. Patent No. 6,123,736, Method and Apparatus for Horizontal Congestion Removal, issued to Pavisic, et al.; and

[0022] U.S. Patent No. 6,070,108, Method and Apparatus for Congestion Driven Placement, issued Andreev, et al.

[0023] The techniques described therein involve the application of cell spreading following the placement step itself. A drawback of this approach is that they do not allow changes in the global qualities of the placement to be ap-

plied to the optimization. Post-placement techniques, such as cell spreading, do not alter the global component of the placement solution. They are valuable for other reasons, as described, e.g., in Patent Application Serial No. 10/063,837 (Attorney Docket BUR920020004US1), "Congestion Mitigation with Logic order Preservation" that falls in this category of algorithm.

[0024] Techniques for optimizing a placement based on timing considerations are described in several patents and can be divided into three categories. The first category generally involves the application of simple timing driven net weighting for placement followed by an in-place optimization (IPO) step. These approaches are limited in their ability to modify the global placement in order to achieve optimization. Examples can be found in:

[0025] U.S. Patent 6,591,407, Method and Apparatus for Interconnect-Driven Optimization of Integrated Circuit Design, issued to Kaufman, et al. ;

[0026] U.S. Patent 6,272,668, Method for Cell Swapping to Improve Pre-layout to Post-layout Timing, issued to Teene;

[0027] U.S. Patent 6,263,478, System and Method for Generating and using Stage-Based Constraints for Timing-Driven Design, issued to Hahn, et al.;

[0028] U.S. Patent 6,192,508, Method for Logic Optimization for Improving Timing and Congestion during Placement in Integrated Circuit Design, issued to Malik, et al.; and

[0029] U.S. Patent 6,523,161, Method to Optimize Net Lists using Simultaneous Placement and Logic Optimization, issued to Gopalakrishnan, et al.

[0030] The second category applies recursive global placement and timing analysis during the forward progression of the algorithm. Examples of these are found in:

[0031] U.S. Patent 6,415,426, Dynamic Weighting and/or Target Zone Analysis in Timing Driven Placement of Cells of an Integrated Circuit Design, issued to Chang, et al.;

[0032] U.S. Patent 6,557,144, Netlist Re-synthesis Program Based on Physical Delay Calculation, issued to Lu, et al.;

[0033] U.S. Patent 6,099,580, Method for Providing Performance-Driven Logic Optimization in and Integrated Circuit Layout Design, issued to Boyle, et al.; and

[0034] U.S. Patent 5,654,898, Timing-Driven Integrated Circuit Layout through Device Sizing, issued to Roetcisoender, et al.

[0035] The third category uses dynamic net weight adjustment and simple logic optimization during placement. Examples are found in: U.S. Patent 6,601,226, Tight Loop

Method of Timing Driven Placement, issued to Hill, et al.;

[0036] U.S. Patent 5,666,290, Interactive Timing-Driven Method of Component Placement that more directly constrains Critical Paths using Net-Based Constraints, issued to Li, et al.; and

[0037] U.S. Patent 6,286,128, Method for Design Optimization using Logical and Physical Information, issued to Pileggi, et al.

[0038] The referenced patents categorized in the three afore-mentioned categories lack the all important "look ahead" characteristic which forms the basis for an efficient optimization, and which includes those related to timing optimization and congestion optimization. The look ahead technique allows the process to closely control the optimization qualities result thereof.

SUMMARY OF INVENTION

[0039] Accordingly, it is an object of the invention to provide a generalized method for optimizing the global placement characteristics of a VLSI chip across multiple cost metrics, such as total wire length, timing, congestion, and signal integrity. The method relies upon a "look ahead" technique, combined with any generic cost function that can be used to set placement directives. These placement di-

rectives include net weights and cell spreading.

[0040] It is another object to provide a method for optimizing congestion that operates within the generalized method.

[0041] It is a further object of the invention to provide a method for optimizing timing that operates within the generalized method.

[0042] The foregoing and other objects are achieved by a method of performing a VLSI placement that involves the iterative reuse of the process of successive partitioning. It is the iterative reuse component of the invention that establishes the capability of "looking ahead" to determine what will happen. Based on the look ahead, it is possible to evaluate the qualities of the placement about to be generated. The method of the present invention proceeds through the placement while maintaining the current state of the placement alongside the look-ahead state of the placement. Based on a continuous evaluation of the look-ahead placement, directives are generated and modified in order that the next steps applied to the current state of the placement will cause it to change to achieve an ultimate higher quality final output.

[0043] This invention decouples the analysis functions / directive generation techniques from the placement flow itself. The

essence of the invention is that it establishes a paradigm for optimizing the placement that allows any generic cost function / directive generation technique to be incorporated therein. This invention also includes two instances of optimization cost functions / directives generation. The first instance is the optimization of timing and the second instance optimization of wiring congestion.

[0044] In another aspect of the invention, there is provided a method for optimizing the placement of a plurality of cells on a VLSI chip that includes the steps of: a) subdividing the plurality of cells into partitions by performing a sequence of cuts; b) iteratively managing the sequence of cuts to perform a look ahead operation; c) returning to the cut from where the look ahead operation was initiated by comparing the original placement with the cut provided by the look ahead operation; and d) altering the priority of the placement to ensure that the quality of the results achieved at the look ahead point is improved.

BRIEF DESCRIPTION OF DRAWINGS

[0045] The accompanying drawings, which are incorporated in and which constitute part of the specification, illustrate presently preferred embodiments of the invention and, together with the general description given above and the

detailed description of the preferred embodiments given below, serve to explain the principles of the invention.

[0046] Figures 1A-1E are pictorial representations of a conventional placement sequence.

[0047] Figure 2 is a flowchart showing the method steps for performing placement optimization relative to a given cost function, in accordance with the present invention.

[0048] Figure 3 is an algorithmic description of the placement flow for a single optimization iteration.

[0049] Figure 4 is an algorithmic description of the placement flow for a generalized number of optimization iterations.

[0050] Figure 5 is an algorithmic description specifically used for congestion removal, in accordance with the invention.

[0051] Figure 6 is a pictorial representation for reserving areas for routability where congestion occurs, with the black stripes showing the blockages created.

[0052] Figure 7 illustrates the algorithm, according to the present invention, as applied to congestion avoidance.

[0053] Figure 8 illustrates the algorithmic description, according to the invention, as applied to timing optimization.

DETAILED DESCRIPTION

[0054] Referring now to Figure 2, there is shown a flowchart illustrating the method steps for performing the placement

optimization relative to a given cost function, in accordance to the present invention.

[0055] The inventive method uses a successive partition placement flow iteratively. In step 2 of the iteration, the flow begins at cut 1 and ends at cut M (step 5). Cut M is determined by the algorithm to be a value such that $M < N$, but sufficiently close to N to allow a placement approximation. Next, the approximated placement from step 2 is evaluated using predefined cost metrics. Based on the evaluation of the cost metrics, directive inputs to placement are specified. This evaluation and directive generation step is shown in steps 8, 9, and 10.

[0056] Still referring to Figure 2, steps 3-5 perform a placement from cut K (step 3) through M (step 5). When the iteration $M > K > 1$ is reached, K is selected so that the starting point of the second iteration captures some global qualities of the initial placement. Likewise, M is computed in accordance with the desired level of placement approximation for the second iteration. Generally, the quality of the placement approximation is expected to increase as the algorithm moves from one step to the next. Steps 3-5 are repeated with the same sequence previously described. For each iteration, the values of K and M increase

and the cost functions are re-evaluated, providing directives to the placement algorithm. At the end of the last step of the placement step, the values from M to N are compared. If M < N, the placement process continues until cut N is performed, at which point, the complete placement is available. The process of successive placements that is based on the refinement of a generalized cost function can be described with reference to Figures 1, 2, 3, and 4. Subsequently, Figures 5 through 8 will show specific applications of the present invention.

[0057] Figure 3 describes the algorithmic implementation that corresponds to the flow chart shown in Figure 2. It is a representation of a special case of the optimization algorithm that limits the number of optimizing iterations to 1. In this special case, the code first performs M cuts (Fig. 3, line 5), in accordance with standard partitioning paradigm described earlier, and depicted with reference to Figure 1. There are two user inputs to the algorithm. The first is K (Fig. 3, line 2), the value of a cut at which the optimizations are to be applied, while looking ahead to a downstream cut. The second user input, shown in line 3, corresponds to the aforementioned look_ahead_factor. This variable is user specified and is set to instruct the applica-

tion to perform the optimizations based on a quantified look ahead value that computes cut M, the exact cut at to which the look-ahead is to occur. The exact mechanics of this operation calls for optimizations to be applied at cut M followed by an immediate restoration of the placement to cut K. Since the intent is to restore the placement to cut K, moving therefrom onward, this sequence will be referred to hereinafter as optimization at cut K with look ahead to cut M (Fig. 3, line 4).

[0058] Figure 4 illustrates a generalized description of the algorithmic corresponding to the flow chart shown in Figure 2. Therein is shown how to generalize the inventive method to any number of optimization iterations. In the most general case, there are four user specified values that control the iteration flow. The first is the look_ahead_factor (Fig. 4, line 3). Just as in Figure 3, this value directs the application to perform optimizations based on a quantified look ahead value that computes cut M. The second input is the forward_increment (Fig. 4, line 4) that computes the values of K and M for each iteration of the algorithm. It is required to be greater than 0. The third user specified value is the max_iterations value (Fig. 4, line 9) that controls the maximum number of iterations

to be performed. In the case of multiple optimization passes, the termination criterion is based on the max_iterations paramater and the value of K (Fig. 4, line 9), which is incremented following each iteration. The fourth user input is C, that pinpoints the value at which the first optimization is to be performed.

[0059] The algorithm proceeds as follows. Starting at cut number 1 the algorithm performs K cuts, where K = C forward_increment (Fig. 4, line 6). At this point, the forward_increment value is deducted from C, but is added back near the end of iteration 1 (Fig. 4, line 16). This causes the iteration 1 to return to iteration 2, i.e., at cut K. Next, the value of M is computed by multiplying N by the user specified look_ahead_value (Fig. 4, line 8). Herein, the main body of the optimization iteration is entered, assuming that the user has specified a max_iterations greater than 0. If the max_iterations value is equal or less than 0, the optimization iteration is skipped, and the remainder of the placement process continues, defaulting to a standard placement algorithm with no optimization possible. In the case where max_iterations > 0, the main body of the optimization iteration is entered (Fig. 4, line 10) and the placement associated with cut K is restored.

On the first time through this loop, the process already stands at cut K, thus nothing happens. At subsequent times through the loop, the restoration process changes the placement accordingly. In line 11, the algorithm performs the look-ahead process by processing cuts K through M. In lines 13, 14, and 15, the algorithm performs a cost function analysis, optimization, and then generates the placement directives. In line 16, the value of K is incremented according to the user specified forward increment. In line 17, K is compared to N (the last cut). If K > N, it is forced to be equal to N, which according to line 8 causes the optimization iteration to terminate. In line 18, M is incremented according to the user specified forward increment. In line 19, M is compared to N. If M > N, it is forced to be equal to N. All the incremental values are required to be greater than zero. The algorithm terminates according to the termination criteria set in line 9, either if the value of K is incremented to N or if the max_ iterations value is reached. The algorithm as specified in the general case performs a number of placement optimization iterations. During the early iterations, the placement optimizations occur at the early cuts, and the look ahead occurs at a cut number that is farther removed

from the last cut. As the iterations proceed, the optimization window moves to cuts that are closer to the end of the placement sequence. In this way, a continuous optimization iteration is achieved resulting in an improvement of the global qualities and local qualities of the placement. The early iterations focus more on the global qualities while the later iterations migrate toward the local qualities of the placement. In this way, a continuous optimization of the placement covering the spectrum from global to local placement optimization is achieved.

[0060] Application of the Present Method to Congestion Optimization. The congestion optimization will now be described with reference to Figure 7. The generic iteration occurs as previously described. The content of the optimization section involves invoking a global router followed by an analysis of the global route results. Based on the analysis placement, directives are generated, including those related to cell spacing and area avoidance. The cell spacing directive is specified by artificially expanding the cell sizes. This is referred to "cell expansion". A blockage is generated to direct the placement algorithm while avoiding placing cells in certain areas. In Figure 7, this is shown by a "blockage insertion".

[0061] The algorithm specification for the flow corresponding to Figure 7 is found in Figure 5. Shown therein is the code used in the top level flow that includes a single optimization iteration, generically documented in the code of Figure 3.

[0062] A description of the congestion optimization flow follows. According to Figure 5, the placement_flow( ) first performs M cuts (Fig. 5, lines 1-2). Next, the global router is invoked, which generates a global routing map. This is well known in the art. The global routing map reflects the degree of wirability of all the areas of the chip. For congestion evaluation purposes, the die area is subdivided into n by n grids. Each grid consists of 30 routing tracks wide and high. Next, the supply (Ws) and the demand (Wd) are computed. The wiring supply on the edge of a grid is computed by:

[0063] Ws = (number of layers)(edge length) blockage, and by power shapes crossing the edge. The wiring demand (Wd) is provided by the global router. Wiring congestion is defined as:

[0064] Cedge= Wd / Ws.

[0065] Finally, the horizontal congestion for the grid is defined as:

[0066] H_grid = (Cedge_left + Cedge_right) / 2,

[0067] wherein Cedge_left and Cedge_right are congestion values associated with the left and right edges, respectively. Similarly, the vertical congestion of the grid is given by:

[0068] V_grid = (Cedge_bottom + Cedge_top) / 2,

[0069] wherein Cedge_bottom and Cedge_top are the values respectively associated with the bottom and top edges. These congestion values determine where a corrective action is necessary and for which placement directives need to be specified.

[0070] Figure 5 shows a technique referred to expand_cells( ). This technique is invoked on cells residing within areas that exceed a user specified congestion value. The result of this action is a placement directive that is generated to force the cells to spread out, thereby modifying the ratio of wiring resource available versus cells that are to be wired. Reducing this ratio directly affects the congestion score by reducing the wiring demand in the local vicinity.

[0071] Although expanding all the cells helps to optimize both the horizontal and vertical congestion, an additional technique is used to specifically address horizontal congestion problems. This technique involves creating placement

blockages in the horizontal congested areas. Examples of these blockages are shown in Figure 6. The horizontal dark stripes identify areas to be left vacant by the placement tool. These are specified by creating a construct known as blockage and forwarding it to the placement tool as part of the directives. The criterion for generating the horizontal blockage strip is similar to what was previously described. The difference is that only the horizontal edge congestion is used. Blockages are introduced based on areas that have horizontal edge congestion that exceeding a user specified value.

[0072] In summary, the steps of global routing, congestion analysis, cell expansion, and blockage generation (Figure 5) are used with the optimization paradigm of this invention (Figure 4) to achieve a placement process that is optimized for congestion.

[0073] Application of the Method for Placement based timing optimization. The application of present invention to the task of a placement based timing optimization can best be described with referred to Figure 8. Therein, a generic optimization flow described in Figures 1, 2, 3 and 4 is once again applied. In the present case, the optimization functions are suited for addressing timing related issues. The

algorithm proceeds as usual by first performing M cuts which correspond to steps 1 through 5 of the flow shown in Figure 8. At this point, an optimization routine is invoked. In the first step (step 6), a timing analysis is performed. Step 7 determines what aspects of the placement fundamentally contribute to the timing problems. In order to achieve this, simple netlist changes, such as repowering, that create timing problems are first removed. Likewise, it is important to buffer long wires and rebuild fanout trees. After performing these operations, the remaining timing problems are either related to the placement characteristics or to the fundamental logic structure. The present optimization corrects the timing problems that are specifically related to the placement of the design. At this point in the flow, step 8 is invoked. Therein, all the nets of the design are sorted according to timing criticality. The nets having the largest criticality are then assigned high net weights. The entire list of nets is traversed, and weights are assigned to each net in proportion to its degree of criticality. The net weights become placement directives for subsequent iterations. The placement algorithm is sensitive to these weights, working hardest on nets that have large net weights. In this man-

ner of directive generation, the placement algorithm is provided with an input that will help to minimize timing problems.

[0074] Following the creation of these directives, the flow of the algorithm returns to step 3, where the placement of the design returns to cut number K, and another cycle of placement of optimization begins. This cycle continues until a termination criterion is reached (see Figure 4, line 9).

[0075] Whereas the present invention has been described in terms of a preferred embodiment, it will be understood by those skilled in the art that numerous changes and modifications to the algorithm may be introduced without departing from the spirit of the invention, all of which fall within the scope of the appended claims.